

This article illustrates various operations performed on SMS.

Contents

- [1 Prerequisite](#)
- [2 Send message](#)
- [3 Read Messaging folders](#)
 - ◆ [3.1 Inbox](#)
 - ◆ [3.2 Outbox](#)
 - ◆ [3.3 Draft](#)
 - ◆ [3.4 Sent Item](#)
- [4 Read incoming message \(Online Mode \)](#)
 - ◆ [4.1 Retrieve Message Body](#)
 - ◆ [4.2 Retrieve Phone Number](#)
- [5 Delete message from Sent Item \(Online Mode \)](#)
- [6 Delete message from Outbox folder \(Online Mode \)](#)
- [7 Delete messages from Messaging Folders](#)
 - ◆ [7.1 Inbox](#)
 - ◆ [7.2 Outbox](#)
 - ◆ [7.3 Draft](#)
 - ◆ [7.4 Sent Item](#)
- [8 Disable "Delivery Report"](#)
- [9 Send message to multiple recipients](#)
- [10 Related Links:](#)

Prerequisite

- Download **SmsHandler.zip** as follows:
 - ◆ For S60 2nd edition = [SmsHander for S60 2nd.Zip](#)
 - ◆ For S60 3rd edition = [SmsHander for S60 3rd.Zip](#)
- Extracting **SmsHandler.zip** will result into **SmsHandler.h** and **SmsHandler.cpp**
- Copy-Paste **SmsHandler.h** into your project's **/inc** folder.
- Copy-Paste **SmsHandler.cpp** into your project's **/src** folder.
- Edit your **.mmp** file. Add an entry for **SmsHandler.cpp** in **SOURCE** directive.

```
SOURCE SMSHandler.      cpp
```

- Edit your **.mmp** file. Add libraries for SMS handling, and capabilities required.

```
//Libraries included for SMS support-
LIBRARY msgs.      lib smcm.lib gsmu.lib mtur.lib
CAPABILITY ReadUserData WriteUserData NetworkServices
```

- Open your **CYrApplicationContainer.h** file.
- Include **SmsHandler.h**.

```
#include "SMSHandler.h" //Added for SMS Handling
```

- Define an object of SmsHandler class.

```
private: //data
        .....
        .....
CSmsHandler* iSmsHandler;
```

- Open your **CYrApplicationContainer.cpp** file.
- Initialize SmsHanlder object.

```
CYrApplicationContainer::ConstructL().....
{
.....
.....
    SetRect(aRect);
    ActivateL();

    iSmsHandler = CSmsHandler::NewL(); // SmsHandler
}
```

Send message

- Define one function of your own SendMsg():

```
void CYrApplicationContainer ::SendMsg()
{
    TBuf<128> SMSText,PhoneNumber;
    SMSText.Copy(_L("Test Message"));
    PhoneNumber.Copy(_L("999999999")); //Replace Number as per your needs

    iSmsHandler->SendL( PhoneNumber, SMSText) ;
}
```

Read Messaging folders

Inbox

Outbox

Draft

Sent Item

- Following code snippet illustrates reading messages from **Inbox Folder**. *KMsvGlobalInBoxIndexEntryId* is used.
- To read messages from Outbox Folder, use *KMsvGlobalOutBoxIndexEntryId*
- To read messages from Draft Folder, use *KMsvDraftEntryId*
- To read messages from Sent Item Folder, use *KMsvSentEntryId*

```

void CSmsHandler::ReadInbox()
{
    HBufC* SMSContent = HBufC::NewLC(400);
    * SMSContent8 = HBufC8::NewLC(400);

    TMsVSelectionOrdering sort
    SetShowInvisibleEntries(ETrue); // we want to handle also the invisible entries

    CMsvEntryContext=CMsvEntry::NewL(*iSession,KMsvGlobalInBoxIndexEntryId,sort); // Reading M
    CleanupStack(inboxContext);

    CMsvEntrySelections = inboxContext->ChildrenL();
    CleanupStack(entries);

    TInt msgCount=entries->Count();
    for (TInt i=0; i<entries->Count(); i++)
    {
        TMsgId entryID;
        ->SwitchCurrentEntryL(entryID);

        * entry=MSMSession->GetEntryL((*entries)[i]);
        ::PushBack(stack);

        * inboxStore=entry->ReadStoreL();
        ::PushBack(stack);
    if (inboxStore->HasBodyTextL())
    {
        TMsVEntry entry1 = entry->Entry();
        <50> aText(entry1.iDetails); // Gives you phone Number or Contact Name if Contact is present
        Copy(aText);          msg.

        // If you want to get the Recipient Number, when iDetails gives you Cont
        iSmsMtm->LoadMessageL();
        CSmsHeader& header = iSmsMtm->SmsHeader();
    }
}

```

SMS_Operations

```
TPtrC from = header.FromAddress(); // This will give you actual phone number

// To read phone number from Sent Item folder, see the Note below

    & richText= iSmsMtm->Body();
    ->RestoreBodyText(&richText);
const TInt length = richText.DocumentLength();

    ->Des().Copy(rSMSContent8.Read(0,length)); // Gives you actual content (Body) of SMS
Reset();      richText.

    ->Des().Copy(SMSContent8>Des());

    (SMSContent8->Des()).WriteToFile("SMSBody.txt"); // Write SMS Body in the SMSBody.txt file
}
else
{
// no text in SMS
}

    ::PopAndDestroy(2,entry);
}

CleanupStackAndDestroy(4, SMSContent);
}
```

Note: To read phone number from the message in the Sent Item folder, use the following code snippet.

```
CSmsHeader& smsHeader = iSmsMtm->SmsHeader();
const CArrayPtrFlat<CSmsNumber>& array= smsHeader.Recipients();
CSmsNumber* smsNumber = array.At(0);
TPtrC RecipientNumber = smsNumber->Address(); // RecipientNumber will contain the recipient Number

//Note that "SMSBody.txt" used in the following code must be created beforehand as we are using O
void CSmsHandler::WriteToFile(const TPtrC& aSMSContent8)
{
    (KFileSpec, "\\SMSBody.txt");//File, in which SMS Body will be stored
    TPos pos;
    RFile fs;
    Connect();
    RFile file;
    if(!file.Open(fs, KFileSpec, EFileWrite);
if(err==KErrNone)
{
    Seek(ESeekEnd, pos);
    Write(aSMSContent8);
    Close(); file.
}
    Close();
//File closed
}
```

Read incoming message (Online Mode)

Retrieve Message Body

Retrieve Phone Number

- You will find **MessageReceivedL()** in the **SMSHandler.cpp** file.
- Perform following changes to read incoming message body and phone number.

```
void CSmsHandler::MessageReceivedL( TMsvId aEntryId )
{
    CMsvEntry entry = iSession->GetEntryL( aEntryId ); // current entry
    CleanupStack( serverEntry );
    TMsvEntry serverEntry = entry->Entry(); // currently handled message entry

    SetNewlyETrue();
    SetIncoming( ETrue );
    SetVisible( ETrue );

    serverEntry->Commit(); // commit changes

//Added to retrieve message body
const TDesC& desc = entry.iDescription; // iDescription will have only first 32 characters from
    <40 bytes message arrived;
    MessageCopy(desc);

//Added to retrieve Phone Number of the Sender
    CSmsMtmCurrentEntryL(aEntryId);
    CSmsMtmMessageL();
    CSmsHeader = iSmsMtm->SmsHeader();

    TPtrC sender.FromAddress();
const TDesC& phoneNumber = from;

    CleanupStackAndDestroy( serverEntry );
}
```

Delete message from Sent Item (Online Mode)

- Add the following case in the **HandleSessionEventL()** of **SmsHandler.cpp**
- So when you send message from your application, it will be deleted from "Sent Item" folder.

```
case EMsvEntriesMoved:
{
// Entry id is obtained from the session event arguments.
    TMsvId* entryId = STATIC_CAST( TMsvId*, aArg2 );

// We are interested in messages that are moved to Sent Item Folder
if ( *entryId == KMsvSentEntryId )
    {
        TMsvSelectionOrdering sort
        sort.SetSorting(EMsvSortByDateReverse);
        SetShowInvisibleEntries(ETrue); // we want to handle also the invisible entries

        * parentCMsvEntryCMsvEntry::NewL(*iSession, KMsvSentEntryId, sort);
        ::PushBack(serverEntry);
    }
}
```

SMS_Operations

```
        CMsvEntrySelect parentEntry->ChildrenL();
        ::PushBack(stack);

for(TInt i = 0; i < entries->Count(); i++)
{
    if( parentEntry->ChildDataL(entries->At(i)).iMtmData3 != KUidMsgTypeSMS.iUid )
    {
        parentEntry->DeleteL(entries->At(i));
        break;
    }
}

::PopBack(stack entries );
::PopBack(stack parentEntry );
}
break;
}
}
```

Delete message from Outbox folder (Online Mode)

- Sometimes a need arises to delete message from the outbox folder so that it cannot be scheduled to be sent.
- Add the following case in the **HandleSessionEventL()** of **SmsHandler.cpp**
- So when you send message from your application, it will be deleted from "Outbox" folder.
- A phone number and the message contents can be retrieved before deleting the message which is shown in the following code snippet.

```
case EMsvEntriesMoved:
{
    // Entry id is obtained from the session event arguments.
    * @MsvId = STATIC_CAST( TMsvId*, aArg2 );

    // We are interested in messages that are moved to Sent Item Folder
    if (*entryId == KMsvGlobalOutBoxIndexEntryId)
    {
        TMsvSelectionOrdering sort
        SetSorting(EMsvSortByDateReverse);
        SetShowInvisibleEntries(ETrue); // we want to handle also the invisible entries

        * parentEntryCMsvEntry::NewL(*iSession,
            KMsvGlobalOutBoxIndexEntryId, sort
            ::PushBack(stack entry);

        CMsvEntrySelect parentEntry->ChildrenL();
        ::PushBack(stack);

for (TInt i = 0; i < entries->Count(); i++)
{
    if (parentEntry->ChildDataL(entries->At(i)).iMtmData3
    != KUidMsgTypeSMS.iUid)
    {
        = entries->At(i);TMsvId entryID
        ->SwitchCurrentEntryL(entries->At(i));
    }
}
```

SMS_Operations

```
->LoadMessageL();          iSmsMtm

    & smsHeader = iSmsMtm->ESmsHeader();
const CArrayPtrFlat<CSmsNumber>& array= smsHeader.Recipients();
    * smsNumber = array.At(0);
    = smsNumberTPAddress; // Gives actual phone number, instead of giving mapped

    & richText= iSmsMtm->BodyRichText
    = richText.DocumentLength;
* SMSContent = HBufC::NewLC(Length);
    ->Des().Copy(richText.BodyContent); // Gives you actual content (Body) of SMS
    Reset();          richText.
// Use SMSContent in your function
    ::PopAndDestroy(SMSContent);
    ->DeleteL(entries->At(parentEntry));
break;
}
}

    ::PopAndDestroy(entries);
    ::PopAndDestroy(parentEntry);
}
break;
}
```

Note: As the message is already deleted before it gets scheduled for actual sending, the code will leave in the *ScheduleL()* function. To prevent showing the error code on the actual device, use TRAP_IGNORE macro around *InvokeAsyncFunctionL* call as shown in the following code.

```
TRAP_IGNORE(iOperation = iSmsMtm->InvokeAsyncFunctionL( ESmsMtmCommandScheduleCopy,
    *selection, dummyParams, iStatus ));
```

Delete messages from Messaging Folders

Inbox

Outbox

Draft

Sent Item

- Following code snippet illustrates deleting messages from **Inbox Folder**. *KMsvGlobalInBoxIndexEntryId* is used.
- To delete messages from Outbox Folder, use *KMsvGlobalOutBoxIndexEntryId*
- To delete messages from Draft Folder, use *KMsvDraftEntryId*
- To delete messages from Sent Item Folder, use *KMsvSentEntryId*

```
void CSmsHandler::DeleteMessages()
```

SMS_Operations

```
{
    TMsgvSelectionOrdering sort
    SetShowInvisibleEntries(ETTrue); // we want to handle also the invisible entries

    CMsvEntryContext=CMsvEntry::NewL(*iSession,KMsvGlobalInBoxIndexEntryId,sort);
    CleanupBackStack(inboxContext);

    CMsvEntrySelections = inboxContext->ChildrenL();
    CleanupBackStack(entries);

    TInt msgCount=msg->Count();
    ; TInt i
for (i=0; i < msgCount; i++)
{
    TMsgvEntryID entryID;
    ->SwitchCurrentEntryL(entryID);

    * entry=CMsvSession->GetEntryL((*entries)[i]);
    ::PushBack(stack

->DeleteL(entryID);
    ::PopBack(stack);
}

CleanupStackAndDestroy(entries);
CleanupStackAndDestroy(inboxContext);
}
```

Disable "Delivery Report"

- Open **SmsHandler.cpp** and add **SetDeliverReport = EFalse** in **CreateMsgL()**

```
TBool CSmsHandler::CreateMsgL()
{
    .....
    .....
    settings->CopyL( iSmsMtm->ServiceSettings() ); // restore settings
    settings ->SetDelivery( ESmsDeliveryImmediately ); // to be delivered immediately
    settings->SetDeliveryReport(EFalse); // Delivery Report Disabled here
    settings->SetServiceSettingsL( *settings ); // new settings
    ....
    ....
}
```

Receiving and deleting received delivery reports is illustrated in [SMS DeliveryReport Deleting Example](#).

Send message to multiple recipients

- Open **SmsHandler.cpp** and set multiple numbers in **AddAddresseeL()** in **CreateMsgL()**

```
TBool CSmsHandler::CreateMsgL()
```

Sent Item

SMS_Operations

```
{
.....
// Recipient number is displayed also as the recipient alias.
    iDetails.Set( iRecipientNumber );

// Add addressee.
    <15>PhoneNumber2;
    PhoneNumber2("9999999999"); //This is second number on which message will be sent
    ->SendMAddresseeL( iRecipientNumber, entry.iDetails );
    ->SendMAddresseeL( PhoneNumber2, entry.iDetails );
.....
.....
}
```

Related Links:

- [Sending SMS with RSendAs](#)
- [SMS Utilities API](#)
- [SMS Receiver](#)
- [Reading SMS from Inbox](#)
- [Sending SMS in S60 3rd Edition - MTM](#)
- [How to send an SMS using sockets](#)
- [Sending-Receiving SMS through an Exe \(Server\)](#)
- [Create Local SMS](#)
- [How to Open SMS or MMS Editor](#)